IONETIX

ADAQAcquisition User's Guide

AIMS Data AcQuisition System

Zachary S. Hartwig

October 10, 2014

Chapter 1

Introduction to ADAQAcquisition

ADAQAcquisition stands for **AIMS D**ata **AcQ**uisition. In general, ADAQAcquisition is a collection of software tools running under Linux[1] for digitizing and analyzing output waveforms from the particle detectors that will be used with Ionetix cyclotron experiments. ADAQAcquisition has been explicitly developed for use with Caen data acquisition hardware and the Caen C libraries that are distributed in support of the various hardware modules. At present, support is provided for a single Caen V6534 high voltage VME board and a signle Caen V1720 digitizerVME board controlled via the Caen V1718 USB-VME board.

1.1 Overview of the ADAQAcquisition software

The centerpiece of ADAQAcquisition is the `ADAQAcquisitionRootGUI` program, which provides an extremely powerful graphical user interface (GUI) for full control of the V6534 and 1720 boards, as well greatly enhanced *digital versions* of the oscilloscope (waveform analysis), multichannel analyzer (pulse spectra creation) and computer programs (graphical plotting, digitial data storage) found in most traditional analog DAQ setups.

`ADAQAcquisitionRootGUI` is primarily bnuilt upon two families of C++ classes. First, `ADAQAcquisitionRootGUI` integrates many classes from the ROOT data analysis toolkit that not only provides the graphical interface libraries themselves but also a tremendously powerful set of data analysis tools, such as graphing, histogramming, curve fitting, peak finding algorithms and much, much more. Second, `ADAQAcquisitionRootGUI` uses two custom C++ classes (ADAQAcquisitionHigh voltage and ADAQAcquisitionDigitizer) that provide a simple but powerful way for the ADAQAcquisition developer to control the V6534 and V1720 boards, obscuring the nitty-gritty of interacting with the boards via the CAEN libraries and exposing easy-to-understand and often significantly enhanced methods for controlling the boards.

Another important component of ADAQAcquisition are the offline waveform analysis tools, which again make full use of the ROOT toolkit. At present, a skeleton template code is provided that will open a specified data file containing waveforms, read in all the

[1] Porting of ADAQAcquisition to Windows/Mac platforms is not foreseen at this time

relevant parameters from the data file, and then iterate through all the stored waveforms in the file. The user is expected to use this template to implement his/her own algorithsm for waveform analysis (counting, pulse spectra creation, pulse shape discrimination, etc).

The purpose of this user's guide is to provide the user of ADAQAcquisition with the information required to

1. Understand the general layout of the ADAQAcquisition source code
2. Retrieve the ADAQAcquisition source code from the Git [2] repository and build the `ADAQAcquisitionRootGUI` executable from source.
3. use `ADAQAcquisitionRootGUI` to acquire and analyze output detector waveforms
4. Begin developing customized offline analysis code to process digital waveform data

To achieve these goals, the remainder of this chapter describes the external software dependencies required to build the ADAQAcquisition source code and run its programs. Chapter 2 provides instructions on how to obtain the ADAQAcquisition source from the Git repository and building `ADAQAcquisitionRootGUI` from source, including configuring the user's Linux environment and some notes on obtaining and building the external software dependencies. Chapter 3 provides a detailed guide to the `ADAQAcquisitionRootGUI` program, including full textual descriptions of each graphical widget available to the user as well as annotated graphics of various parts of the program. Chapters 4 and 5 contain step-by-step tutorials that walk the user through some of the most important features of the `ADAQAcquisitionRootGUI` program. Finally, Chapter **??** provides an overview of the `ADAQAcquisitionRootGUI` source code to enable the ambitious user to add his/her own improvements to the software.

1.2 Software Dependencies

This section briefly describes the external software dependencies required to build and execute the ADAQAcquisition source code.

1.2.1 The CAEN libraries

CAEN distributes hardware drivers and C libraries in support of its data acquisition system hardware. The following is a complete list of the required CAEN libraries for ADAQAcquisition . Note that the most up-to-date version of the CAEN libraries and driver that *have been successfully tested* with ADAQAcquisition are listed in parenthesis next to the library name:

1. **CAENVMELib** (*CAENVMELib-2.30*): provides a set of C functions that enable communication with the VME crate backplane and attached VME modules with the CAEN "bridge" modules, such as the VME-USB V1718 board.
2. **CAENComm** (*CAENComm-1.02*): provides a set of relatively high-level C functions that mask the low-level details of communication with VME hardware via VME protocol, such as reading/writing registers and other common VME functionality.

[2] Subversion code management system. Available at: `http://subversion.apache.org`

3. **CAENDigitizer** (*CAENDigitizer-1.31*): provides a set of relatively high-level C functions that mask the low-level details of communication with the CAEN family of digitizers. Explicitly meant for digitial data acquisition with CAEN sampling ADC (analog-to-digital) modules such as the V1720 board.

4. **V1718 Linux Driver** (*Version 0.9*): provides necessary Linux driver to interface with the VME-USB V1718 board. Must be built from source and installed before using `ADAQAcquisitionRootGUI`.

The CAEN dependencies listed above are provided as part of the ADAQACQUISITION source code distribution for convenience and to ensure explicit version of control of the various libraries.

The CAEN homepage: `http://www.CAEN.com`

1.2.2 The ROOT toolkit

ROOT is a free (as in no cost), open-source, C++ object-oriented data analysis toolkit that was originated at CERN but is now developed and maintained by a worldwide collaboration across many scientific fields, although it is primarily used in particle and nuclear physics.

While it was developed explicitly for handling massive data sets, it is also incredibly useful for data analysis on a small scale, providing powerful histogramming, fitting, graphical output, and GUI generation. It also contains a diverse array of powerful data analysis tools that are available as class objects that can be applied to any data set of data regardless of size. ROOT is also powerful because the user can fold his or her own C++ code with the ROOT class objects to create highly dynamic and customized data analysis, using all of the power of C++: the standard libraries, the BOOST libraries, etc. It is open-source, constantly improved and maintained, and there is a dynamic user community available online.

The ROOT homepage: `http://root.cern.ch/drupal/`

The ROOTTALK users forum: `http://root.cern.ch/phpBB2/`

1.2.3 The Boost C++ libraries

BOOST is a set of free (as in no cost), open-source, peer-reviewed portable C++ libraries that are incredibly powerful and diverse. BOOST can be thought of as a significant extension of the C++ Standard Libraries and is often a testing ground for a majority of the code that eventually gets folded into C++ during technical reviews.

The BOOST homepage: `http://www.boost.org`

Chapter 2

Building CyDAQ from the SVN Repository

This chapter describes in detail the steps required to obtain the ADAQAcquisition source code from its SVN repository home and then build the ADAQAcquisition executables from source.

2.1 Prerequisites

Before building the ADAQAcquisition source code, a number of dependencies must be installed on the user's local system in order to ensure a successful build of the ADAQAcquisition source code. While the specific details of building each of these dependencies is outside the scope of this manual, a few notes are provided below. It should be noted that all required software dependencies are already installed on the Ionetix server, greatly accelerating the process of building and the using the ADAQAcquisition software.

2.1.1 ROOT

If the user will be building ADAQAcquisition on his/her local Linux machine, ROOT may be downloaded from the following location: `http://root.cern.ch/drupal/content/downloading-root`. The most recent version of ROOT that has been successfully tested with the ADAQAcquisition source code is version 5.32.01. While ROOT binaries are available for direct installation, it is preferable to build ROOT from source code to ensure perfect configuration to the user's local environment. The user is recommended to download the appropriate tarball and then follow the directions for installing ROOT from source that can be found here: `http://root.cern.ch/drupal/content/installing-root-source`

If the user will be building ADAQAcquisition on the Ionetix server, ROOT version 5.32.01 has already been installed in `/usr/local/root/root_v5.32.01`. To use this version of ROOT , the user needs to add the following lines to his/her `.bashrc` file:

```
export ROOTSYS=/usr/local/root/root_v5.32.01/root
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
```

These environmental variables ensure that the user has full access to the ROOT binaries and libraries required to build and run the ADAQAcquisition source code.

2.1.2 Boost

If the user will be building ADAQAcquisition on his/her local Linux machine, the Boost C++ libraries can either be installed via the operating systems package manager (`apt-get` on Ubuntu or `yum` on Fedora/RedHat) or downloaded from the Boost homepage at `http://www.boost.org/`. The vast majority of Boost functionality is conventiently provided by header files and therefore does not require building from source; such is the case for the parts of Boost used by ADAQAcquisition . The user can follow directions for installation of the most recent Boost release (1.49.0) on Linux machines here: `http://www.boost.org/doc/libs/1_49_0/more/getting_started/unix-variants.html`

if the user will be building ADAQAcquisition on the Ionetix server, `Boost` is already installed at `/usr/include/boost`. No special action is required of the user; `Boost` is ready to be used.

2.2 Getting the ADAQAcquisition source code

The ADAQAcquisition source lives permanently in a Subversion code management repository on the Ionetic Linux server [1]. While anyone with a user account on the Ionetix Linux server may check out a working copy of the code, only users who are in the "CyDAQ" group have permission to check in changes to the repository. To be added to the CyDAQ developers group, please contact the author at `hartwig@psfc.mit.edu`.

Checking out a copy of the source code depends on the location from which you connecting to the SVN repository: a "local" connection refers to a user who is logged into and working on the Ionetix server; a "remote" connection refers to a user who is working on a remote computer with SSH capababilites that is running Subversion.

To checkout a copy of the ADAQAcquisition source code *locally*, the user should open a terminal on the Ionetix server and type:

```
svn co file:///usr/local/svn_repos/_CYDAQ_REPO_ /path/to/checkout
```

where the user should replace `/path/to/checkout` with the directory he/she would like SVN to place the checked out ADAQAcquisition source code.

To checkout a copy of the ADAQAcquisition source code *remotely*, the user should open a terminal on his/her local machine and type:

```
svn co svn+ssh://username@ipaddress/usr/local/svn_repos/_CYDAQ_REPO_ /path/to/checkout
```

where the user should replace `username` with his/her username on the Ionetix server, `ipaddress` with the IP address of the Ionetix server, and /path/to/checkout with the directory on the user's local machine that he/she would like SVN to place the checked out ADAQAcquisition source code.

[1] For the uninitiated, a good code management system is a highly scalable piece of software that allows developers to efficiently, seamlessly, and confidently work on a software project together with the full history of everyone's changes to the project (as well as all of the original files) fully documented and accessible at any time. They are incredibly useful for a developers or even just a single developer, if for nothing other than to document all changes and provide the ability to go back to any version of any file at any time.

2.3 Overview of ADAQAcquisition directory structure

After checking out the ADAQAcquisition source code from the SVN repository as described in Section 2.2, navigate to the top-level ADAQAcquisition directory, which will be the path specified to SVN when the copy of the source code was checked our of the repository (/path/to/checkout in the directions above). This directory is known as the top-level directory. The directory and file structure below this directory is descibed below:

- **analysis/**: contains a "template" for offline data analysis of digitized waveforms produced with `CyDAQRootGUI`. Designed to show the user how to access stored waveforms and provide a basis for his/her own analysis code.
- **manual/**: contains all of the files (LaTeXfiles, images files in the images/ subdirectory, and the GNUmakefile for creating the beautiful manual you are now reading.
- **source/**: contains the bulk of the ADAQAcquisition source code. The **versions/** subdirectory (empty as present) will contain snapshots of the ADAQAcquisition source. The snapshots will be versioned and used when acquiring mission critical data, such that the state of the ADAQAcquisition source code can be linked to experimental data sessions. The **trunk/** subdirectory contains the `CyDAQRootGUI` source code in subdirectories and the GNU makefile:
  - **bin/**: contains the `CyDAQRootGUI` binary after building via the GNU makefile. Other useful binaries are also in this directory.
  - **build/**: a dumping ground for transient files created during the build process. The files are source code object files and ROOT dictionary files.
  - **include/**: C++ header files for `CyDAQRootGUI`.
  - **lib/**: CAEN libraries required to build and run `CyDAQRootGUI`. The purpose of including these libraries here (rather than relying on installed CAEN libraries in places like /usr/lib) is for portability and to maintain explicit control over the versioning of CAEN libraries. The CAEN libraries are evolving and only somewhat decently documented, so it is important to ensure the correct versions of the libraries are used. Subdirectories contain libraries and soft links for both 32- and 64-bit architectures in the **x86/** and **x86_64/** subdirectories.
  - **src/**: C++ source code for `CyDAQRootGUI`.
- **utilities/**: contains some useful Python scripts.

With the ADAQAcquisition source code checked out and a basic understanding of the source code structure, it's time to build the `CyDAQRootGUI` binary.

2.4 Configuring your environment

The ADAQAcquisition source requires several environmental variables to be defined in order to build and run correctly.

- **CYDAQHOME**: Used by the ADAQAcquisition GNU makefiles to determine the full paths to a number of source files and libraries. The value should be set as the full path to the top level of the ADAQAcquisition source code.

- **HOSTTYPE**: Used by the ADAQAcquisition GNU makefiles to build the ADAQAcquisition software tools for the correct architecture. The value should be set to "x86" for 32-bit architecture and "x86_64" for 64-bit architectures.
- **PATH**: The Linux environmental variable that contains directories that are searched to determine the location of binaries. The bin directory for CyDAQRootGUI should be added to the `PATH` variable.
- **LD_LIBRARY_PATH**: The Linux environmental variables that contains directories that are searched for dynamically linked libraries. The ADAQAcquisition source code contains most of the required libraries for both 32- and 64-bit architectures. The lib directory for CyDAQRootGUI should be added to the `LD_LIBRARY_PATH` variable.

The easiest way to set these four variables is to add the required lines to the user's `.bashrc` file (which exists in his/her home directory). For example, consider that user `zaphod` has checked out a copy of the ADAQAcquisition source code to the directory `/home/zaphod/CyDAQ_CheckOut` of the Ionetix server, which runs a 64-bit operating system. He would need to add the following lines to the file `/home/zaphod/.bashrc`:

```
export CYDAQHOME = /home/zaphod/CyDAQ_CheckOut
export PATH = $CYDAQHOME/source/trunk/bin:$PATH
export HOSTTYPE = x86_64
export LD_LIBRARY_PATH = $CYDAQHOME/source/trunk/lib/$HOSTTYPE:$LD_LIBRARY_PATH
$
```

After these lines are added to the `.bashrc` file, the user will either need to source the file in the currently open terminal or simply close the terminal window and reopen a new one for the environmental variables to take effect.

2.5 Building the `CyDAQRootGUI` binary

After ensuring that the prequisites described in Section 2.1 have been successfully installed and that the user's environment is configured as described in Section 2.4, the user is ready to build the `CyDAQRootGUI` binary.

- Navigate to the `CyDAQRootGUI` directory:

  ```
  cd CYDAQHOME/source/trunk
  ```

- Build the binary from the source code using the provided GNU makefile:

  ```
  make
  ```

  The GNU makefill will show the compilation steps and update the user on the progress of the build.

- If successful, a new executable binary named `CyDAQRootGUI` should exist in the `$CYDAQHOME/source/trunk/bin` directory

That's all it takes! The user is now ready to begin running the `CyDAQRootGUI` and should proceed to Chapter **??**.

Chapter 3

An overview of ADAQAcquisition

This chapter gives a detailed overview of the ADAQAcquisition program, which provides a rich, graphical environment for simple but powerful control over the entire data acquisition system. The user will find directions to start the program, as well as complete textual descriptions of each widget's functions and inputs. Annotated images of the various frames of ADAQAcquisition to demonstrate the layout and location of all the widgets is also provided.

3.1 Getting started

Because the directory containing the ADAQAcquisition binary should be in the user's `PATH` environmental variable (See Section 2.4), the user may simply type:

```
CyDAQRootGUI
```

from any directory to start the program. Note that the directory from which the user chooses to execute CyDAQRootGUI will, by default, receive any output files (data files, graphic files) that the user produces while running the program. Therefore, the user should at least ensure that they have write access to the execution directory. If all goes well, the program GUI should appear on the screen.

3.2 Description of ADAQAcquisition widgets

This section contains detailed textual and graphical descriptions of the widgets contained in the program.ADAQAcquisition is divided into three main windows, or "frames", with labelled tabs appearing in the upper-left corner of the main ADAQAcquisition window. The user may switch to a particular frame by clicking on the associated tab. Each main frame contains functionality that is logically grouped under the appropriate tab heading. The following subsections describe each widget contained on the main frames in detail and visually show the main layout of each frame.

3.2.1 The VMEConnection Frame

The VMEConnection frame contains basic VME interaction functions, such as establishing a VME connection to the VME boards, setting the VME boards addresses in VME space, as well as reading/writing individual registers from the V1720 and V6534 boards. An annotated view of the VMEConnection frame appears in Figure 3.1

The following list contains a description of the function and input required for each widget that appears on the VMEConnection frame:

- **Connect/Disconnect button**: a large rectangular button that appears at the top-center of the frame initially as red. Clicking the button will attempt to connect to the V1720 and V6534 VME boards at the VME address specified in each board's VME address widget. If the connection is successful, the button will turn green and the text will indicate that the VME connection has been established. Clicking the button while the connection is estblished (the button is green) will then attempt to close the VME connection, returning the button color to red and setting the text to indicate the VME connectinon has successfully closed.

- **V6534BaseAddress number entry**: The 32-bit address of the V6534 board in VME space must be entered as an 8 digit hex number, with the first 4 hex digits corresponding to the VME address potentiometer settings on the physical V6534 board.

- **V6534 Read Cycle**: widgets that are used to manually read a single 16-bit value from one of the registers on the V6534 board and display it for inspection.

  - **Offset address number entry**: a 4 digit hex number describing the V6534 register address from which the register's value will be read.
  - **Value**: the value stored at the register address specified in the address number entry above
  - **VME read button**: initiate the VME read cycle to obtain the value from the V6534 register address specified in the address number entry above.

- **V6534 Write Cycle**: widgets that are used to manually write a single 16-bit value to one of the registers on the V6534 board. Error checking is performed internally to prevent overwriting restricted registers.

  - **Offset address number entry**: a 4 digit hex number describing the V6534 register address to which the register's value will be written.
  - **Value**: the value that will be written to the address specified in the address number entry above
  - **VME read button**: initiate the VME write cycle to write the value to the V6534 register address specified in the address number entry above.

- **V1720BaseAddress number entry**: The 32-bit address of the V1720 board in VME space must be entered as an 8 digit hex number, with the first 4 hex digits corresponding to the VME address potentiometer settings on the physical V1720 board. It should be noted that if the user desires to use the CAEN DT5720 desktop digitizer in place of the V1720 board then the VME address should be set to 0x00000000. The DT5720 should then be connected directly to the Ionetix server via the USB cable, bypassing the VME8100 crate and V1718 USB-VME board completely.

- **V1720 Read Cycle**: widgets that are used to manually read a single 32-bit value from one of the registers on the V1720 board and display it for inspection.
  - **Offset address number entry**: a 4 digit hex number describing the V1720 register address from which the register's value will be read.
  - **Value**: the value stored at the register address specified in the address number entry above
  - **VME read button**: initiate the VME read cycle to obtain the value from the V1720 register address specified in the address number entry above.
- **V1720 Write Cycle**: widgets that are used to manually write a single 32-bit value to one of the registers on the V1720 board. Error checking is performed internally to prevent overwriting restricted registers.
  - **Offset address number entry**: a 4 digit hex number describing the V1720 register address to which the register's value will be written.
  - **Value**: the value that will be written to the address specified in the address number entry above
  - **VME read button**: initiate the VME write cycle to write the value to the V1720 register address specified in the address number entry above.

3.2.2 The High Voltage Frame

The High Voltage frame provides full control over the settings of the V6534 high voltage board, including the voltage, current, and power settings for each of the 6 channels. Real-time monitoring of set voltage, drawn current, and power state ("on" or "off") is provided. An annotated view of the High Voltage frame appears in Figure 3.2.

- **High voltage settings for Channels 0–6**: There are 6 identical "group frames" (with the exception of the V6534 channel they refer to as indicated by their label) that encapsulate the widgets for control of each channel's high voltage settings.

  At present, a channel's set voltage and maximum drawn current can be set only while the channel is powered off, i.e. "live" changes to the voltage and current settings are not allowed. To modify an energized channel's voltage and current, the user must power the channel off, reset the voltage and current levels, and then power the channel back up. Automatic disabling the of voltage and current number entry widgets when a channel is energized enforce this standard. In the future, real-time adjustments of a channel's voltage and current while the channel is energized may be enabled.
  - **Set Voltage number entry**: the channel's voltage setting. Valid integer values are between 0 and 6000. Units of the number entry are in volts (V).
  - **Set Current number entry**: the channel's maximum allowed drawn current. If more current is drawn that is set in this widget, the channel will trip off. Valid integer values are between 0 and 1000. Units of the number entry are in microamps (μA).

  - **Active Voltage and Active Current number displays**: Disabled number entry widgets to the right of the "Set" number entry widgets that display the channel's current set voltage and drawn current when the "Enable monitoring" check button is set (See below).
  - **ON/OFF power button**: Button to the right of the channel set and display numebr entries. Button is red and marked "OFF" when the channel power state is "OFF". When clicked, the button will turn "green" and the text will be changed to "ON" to indicate that the channel has been energized. Note that the voltage/current ramp up and down relatively slowly and changes

- **Enable Monitoring check box**: When this button is checked, the present values of active voltage and drawn current for all 6 channels are displayed in the "Active Voltage" and "Active Current" disabled number entry widgets for each channel. Updates are provided every 1 second.

3.2.3 The Oscilloscope frame

The Oscilloscope frame contains the powerful core of the ADAQAcquisition binary. Much more than the simple oscilloscope the frame name implies, this frame contains a full feature digital oscilloscope, widely configurable multichannel analyzer, graphical plotter, and persistent data storage engine. The term "DGScope" (digitizer scope) is used to describe this powerful analysis tool. An annotated view of the entire Oscilloscope frame appears in Figure 3.3, while close-up views of the important subframes appear in Figures 3.4–3.8.

The Oscilloscope frame is broadly divided into three main subframes. The narrow, vertical subframe (the "channel subframe") at the leftmost side of the main frame contains 8 group frames (corresponding to the 8 V1720 digitizer channels). Each group frame (labelled by channel number from 0 to 7) contains 7 widgets that control channel-specific settings. A scroll bar is provided at the right of the subframeis provided to access to all 8 group frames and their encapsulated widgets. An annotated picture of a single channel group frame containing 7 channel-specific widgets appears in Figure 3.4.

The large, almost square subframe (the "canvas subframe") at the top-right of the main frame contains four widgets: a large canvas for plotting digitized waveforms and pulse spectra; a double vertical slider for Y axis zoom control; a triple vertical slider for X axis zoom control and calibration; and a large, red button labelled "Stopped" by default for starting/stopping data acquisition. The canvas subframe can be easily seen in Figure **??**.

The final subframe (the "scope subframe") sits underneath the canvas subframe at the bottom-right of the main Oscilloscope frame. The scope subframe holds 4 tabbed subsubframes (Thins are etting complicated, I know...), all of which hold widgets that apply general settings of the DGScope: global scope function settings, pulse spectrum settings, graphical settings, and data settings. Annotated pictures of each of the 4 tabbed subsubframes can be seen in Figures 3.5–3.8

- **Channel Group Frame**: The 7 widgets contained in the channel group frame control channel-specific settings.
  - **Enable check button**: Enable/disable the digitizer channel. When enabled, signals input into the corresponding V1720 channel will be digitized, waveforms will be displayed in "Waveform" mode, and the waveforms will be processed into pulse spectra.

- **Pulse polarity radio buttons**: Used to indicate the direction of waveform rise relative to the signal baseline."+" indicates that the pulse rises above or to the positive side on the Y axis of the X axis baseline; "-" indicates that the pulse rises below or to the negative side on the Y axis of the X axis baseline. While not necessary for plotting waveforms, pulse polarity is critical to the algorithms used to in the creation of pulse spectra.
- **Vert. Position number entry**: Sets the vertical position of the baseline relative to it's "0" position, i.e. a plotted offset of the input DC level of the baseline. This setting merely applies to the post-digitized *plotted* waveforms and does not affect settings such as the trigger threshold, although the plotted trigger line will shift along with the channel baseline for consistency. This setting is primarily used to view multiple digitized waveforms simultaneously on the same DGScope canvas. Units of the settings are in ADC (analog-to-digital) units.
- **DC offset number entry**: Applies a DC offset to the input signal *before* digitization such that the dynamic input range of the digitizer channel may be swung from -2–0 volts to 0–+2 volts. The input values are 16-bit integers in hex, such that 0x0000 corresponds to -2–0 volts, 0x8000 corresponds to -1–+1 volt, and 0xffff corresponds to 0–+2 volts. For details, please see the CAEN V1720 Digitizer Manual, Section XX, Page XX.
- **Trigger threshold number entry**: Sets the channel trigger threshold in ADC units. Waveforms that exceed this trigger will cause a global (all 8 channel) trigger-data acquisition cycle of the V1720 board when DGScope triggering is set to run in "automatic" mode.
- **Baseline min. and max. number entry**: Sets the range to use (from the minimum sample number to the maximum sample number) that is used for compute the average baseline value of the waveform. The range will be plotted on the DGScope when in "Waveform" mode as a shaded box in each channel's colors over the range used to calculate the baseline.

• **The Canvas Subframe**

- **Double vertical slider**: The double vertical slider is used to control the Y axis zoom when DGScope is in either "Waveform" or "Spectrum" mode. The user may drag either end of the slider to zoom in or out, as well as dragging the entire slider by clicking-and-holding in the center of the slider to translate the view along the Y axis.
- **Triple vertical slider**; The triple vertical slider is used to control the X axis zoom when DGScope is in either "Waveform" or "Spectrum" mode. The user may drag either end of the slider to zoom in or out, as well as dragging the entire slider by clicking-and-holding in the center of the slider to translate the view along the X axis. In addition, the third slider tab that moves along the width of the slider body is used during calibration when DGScope is in "Spectrum" mode. For details on calibration, see Section **??** of this manual.
- **Acquisition button**: the long button immediately underneath the X axis triple slider control the starting/stopping of data acquisition using DGScope. Initially red with text "Stopped", clicking the button will turn the button green and update the text to "Acquiring", indicating that DGScope is now acquiring and

processing waveforms on the enabled digitizer channels. Clicking the button again will stop data acquisition and return the button to it's original state.

- **The Scope Controls subsubframe**: contains widgets that control global functionality of DGScope:
  - **DGScope Mode radio buttons**: sets the global function of DGScope. In "Waveform" mode, DGScope acts like a fully digital oscilloscope for visualization of input waveform from all enabled channels. In "Spectrum" mode, DGScope acts like an enhanced, fully digitial multichannel analyzer (MCA) plus computer, facilitating pulse spectra creation with different pulse processing algoriths. In "Blank" mode, nothing is plotted in DGScope, greatly reducing the CPU overhead of DGScope and making this mode extremely efficient for high-throughput data acquisition.
  - **Coincidence Triggering check button**: when checked and DGscope is running in "automatic" triggering mode, a trigger is only generated if 2 or more channels have exceeded their respective channel trigger threshold and the waveforms overlap in time. For more details, see the CAEN V1720 digitizer manual, Section XX, Page XX.
  - **Coincidence Level selection box**: allows setting the coincidence level, i.e., the number of channels that must all exceed their respective trigger theshold to enable a trigger when DGScope is running in "automatic" mode and the "coincidence trigger" check button is enabled.
  - **Trigger Type selection box**: allows setting the V1720 digiter to trigger in one of 3 modes. "External" triggering mode generates global triggers on the rise of an input timing pulse, using either NIM fast logic or TTL fast logic signals. The NIM/TTL pulse must be fed into the external triggering input on the V1720 digitizer front panel via a LEMO 00 connector on 50Ω. "Automatic" triggering mode generates global triggers when one channel (or multiple channels if using coincidence triggering) exceeds its set trigger threshold. "Software" triggering mode generates a global trigger when the user clicks the manual trigger button.
  - **Manual Trigger button**: Clicking the button generates a software trigger. When the triggering mode is set to "software", a global trigger is generated on the V1720 digitizer.

- **The Spectrum Settings subframe**: contains widgets that control the behavior of the pulse spectra:
  - **Histogram channel selection box**: Select which enabled channel's pulse spectra histogram will be plotted in the DGScope canvas. At present, only one channel can be plotted at a time, although the user may switch between channels during acquisition.
  - **Number of bins number entry**: Select the number of bins used in the pulse spectra histogram
  - **Mininmum and maximum bin number entries**: Set the minimum and maximum bin values of the pulse spectra histogram

- **PHS and PAS radio buttons**: Select whether pulse height (PH) or pulse area (PA) algorithms will be used to produced pulse spectra (S) from digitized waveforms.
- **LLD and ULD number entries**: Set the lower level discriminator (LLD) and upper level discriminator (ULD) values. Waveforms with pulse height/area that are below the LLD or above the ULD are not histogrammed into the spectrum.
- **Use LD trigger check box**: Use the LLD/ULD values from one channel (see next widget description) as a "trigger" for saving digitizer waveforms to an open ROOT file. The purpose is to "trigger" only on pulses between known pulse (energy) bins.
- **LD trigger selection box**: Select the channel that will be used for level discriminator (LD) triggering. Note that when this channel triggers, i.e. that analyzed waveform falls between the LLD and ULD bin values, a global trigger is generated that triggers all 8 V1720 digitizer channels.
- **Calibration mode check box**: When checked and DGScope is running in "spectrum" mode, the user may calibrate the pulse spectrum to convert the X axis of the spectrum from "pulse units" in ADC to "energy units" such as keV.
- **Calibration point selection box**: Selects the calibration point to be set. The number of selection options starts at 0 and increases by 1 each time the user adds a point to the calibration by clicking the "Set Point" button.
- **Calibration energy number entry**: Enter the known particle energy in keV corresponding to a peak in the present spectrum, which has the third horizontal slider tab centered on it or which will have its pulse unit entered in the "Pulse Unit" number entry.
- **Pulse unit number entry**: Should be set to the center value of a peak in the pulse spectrum that represents the known particle energy entered in the "Calibration energy" number entry widget. The pulse unit will be automatically set (and this number entry updated) when the user drags the third horizontal slider tab to center the vertical dashed red line (which will appear on the DGscope canvas) on the peak in the spectrum.
- **Set point button**: Sets the current calibration point selected in the Calibration Point selection box. Clicking the button assigns the energy and pulse unit values to a point on the calibration curve, which is an energy vs. pulse unit curve for converting the pulse spectrum to an energy spectrum. After this button is clicked, a new calibration point will appear in the Calibration Point to continue adding points to the calibration curve.
- **Calibrate button**: Uses all of the set calibration points to create a calibration curve. Once this button is clicked and the Calibration Mode check box in unchecked, the calibration curve is used to histogram all analyzed waveforms (pulse height or pulse area values in ADC) into an energy spectrum, where the bin values now represent particle energy in keV.

- **Graphical Options**: contains widgets that control the appearance of what is plotted on the DGScope canvas:
  - **X-axis title text entry**: Sets the title of the X-axis for both the waveform and spectrum plot.

- **X-axis title offset text entry**: Sets the position of the X-axis title below the X-axis labels.
- **Y-axis title text entry**: Sets the title of the Y-axis for both the waveform and spectrum plot.
- **Y-axis title offset text entry**: Sets the position of the Y-axis title below the Y-axis labels.
- **Graph title text entry**: Sets the title of the DGScope graph.
- **Draw legend check box**: If checked, draws a legend on the waveform plot showing the channel-to-color mapping.
- **Log. X-axis / Log. Y-axis check boxes**: If checked, will display the X-axis / Y-axis in logarithmic scale. Can be freely switched during data taking.
- **Waveform X-axis radio buttons**: Will display the X-axis for waveform plotting in units of Sample (recall that a sample is taken every 4 nanoseconds with the V1720 digitizer) or nanoseconds ("ns"). Can only be set when DGScope is not acquiring data.
- **Waveform Y-axis radio buttons**: Will display the Y-axis for waveform plotting in units of ADC (integer analog-to-digital units, 0 being the minimum and 4095 begin the maximum) or millivolts ("mV"). Can only be set when DGScope is not acquiring data.
- **Output file name text entry**: Sets the name of the image file that will contain the current contents of the DGScope canvas.
- **Output file type selection box**: Sets the file type for the image file: .eps, .ps, .png, .jpeg image formats are current supported. But please, be a professional and use .eps. Vector graphics are the only way to go! No one wants to see a rastered image in a professional report.
- **Save plot button**: Saves the contents of the DGScope canvas as they currently appear to the file name and format set in the previous widgets. Can be clicked while DGScope is acquiring data. In addition, all file output names end in an integer that is incremented if the user attempts to create image files of the same name.

- **Data Storage**: contains widgets that control persistent storage of digitized waveforms in ROOT files:
  - **Filename text entry**: Sets the name of the ROOT file to be opened for receiving digitized waveforms. The name should (by convention) end in the .root extension to ease file identification for yourself and other users.
  - **Comment text entry**: Allows the user to entry anything he/she wants to be stored in the ROOT file. Entries could describe the current measurement, parameters, notes, ideas, or nice compliments about the author of this manual.
  - **Create ROOT file button**: When DGScope is acquiring data and this button is clicked, the ROOT file (name set in the previous widget) will be created on the hard drive. This widget is disabled when DGScope is not acquiring data. Note that an error message will be printed to the terminal from which ADAQAcquisition was executed if the user tries to create a ROOT file with the same name

as an existing ROOT file. Note also that, while the file is opened and some measurement data has been written to the ROOT file, waveforms are not stored in the file until the Data storage check button is engaged.

- **Close ROOT file button**: Closes the currently open ROOT file. Before the file closes, all required ROOT operations are performed to ensure all waveform data and structures within the file are written correctly.
- **Data stored when checked button**: When this button is checked, waveforms are written to the currently open ROOT file.

3.3 Persistent data storage in ROOT files

This section describes how ADAQAcquisition stores data in ROOT files, which is critical to understand how to use offline analysis codes to extract the data from the ROOT files and analyze it. An understanding of C++ programming and object-orienting principles are helpful but not required to grasp the essentials of this section.

3.3.1 The ROOT file

A complete description of what a ROOT file is outside of the scope of this manual[1]; however, a simple description is that a ROOT file is like a unix directory, in that it can contain unlimited further directories, files, objects, etc, all in a machine-independent format for portability.

The ROOT file format created by ADAQAcquisition is relatively simple and contains only three objects:

- A class object of type `CyDAQRootMeasParams`, which is a custom-defined C++ class that contains the value of each channel's settings during a measurement and the RecordLength (acquisition window width) as member data. The class declaration for `CyDAQRootMeasParams` is:

```
1   class CyDAQRootMeasParams : public TObject{
2
3     public:
4       std::vector<double> DetectorVoltage;
5       std::vector<double> DetectorCurrent;
6       std::vector<double> DCOffset;
7       std::vector<double> TriggerThreshold;
8       std::vector<double> BaselineCalcMin;
9       std::vector<double> BaselineCalcMax;
10
11      int RecordLength;
12
13      ClassDef(CyDAQRootMeasParams,1);
14  };
```

As can be seen on line 1, the class inherits from a ROOT `TObject` class, which helps it seamlessly interface with the ROOT file and ROOT offline data analysis code. Lines 4–9 show that one C++ standard library `vector` container is used to store a single type of channel parameter; thus, each vector will have 8 entries corresponding to

[1] The interested user can find all the gory details in chapter 11 of the ROOT manual: `http://root.cern.ch/download/doc/ROOTUsersGuideHTML/ch11.html`

the 8 V1720 digitizer channels. Line 11 shows the RecordLength is simply an integer. Finally, Line 13 simply declares the class to ROOT .

- a class object of type `TObjString`, which is a ROOT class that represents a string. It is used to store whatever additional information the user would like to have attached to the data contained in the ROOT file as a string.

- A ROOT `TTree` that contains all of the digitized waveforms. A `TTree` is essentialy a hierarchical structure for logically categorizing and efficiently storing large quantities of data. Here, the tree is subdivided into 8 "branches" (one for each V1720 digitizer channel) that holds the digitized waveforms as C++ standard library `vector` objects of length "RecordLength". Theoretically, an unlimited number of `vector` objects representing digitized waveforms can be stored on each channel's branch.

While ADAQAcquisition is actively acquiring data, the user has the option to create a new ROOT file on the "Data Storage" subframe on the "Oscilloscope" frame. The act of clicking the "Create ROOT File" button performs the following actions:

- creates the user-named ROOT file on the harddisk

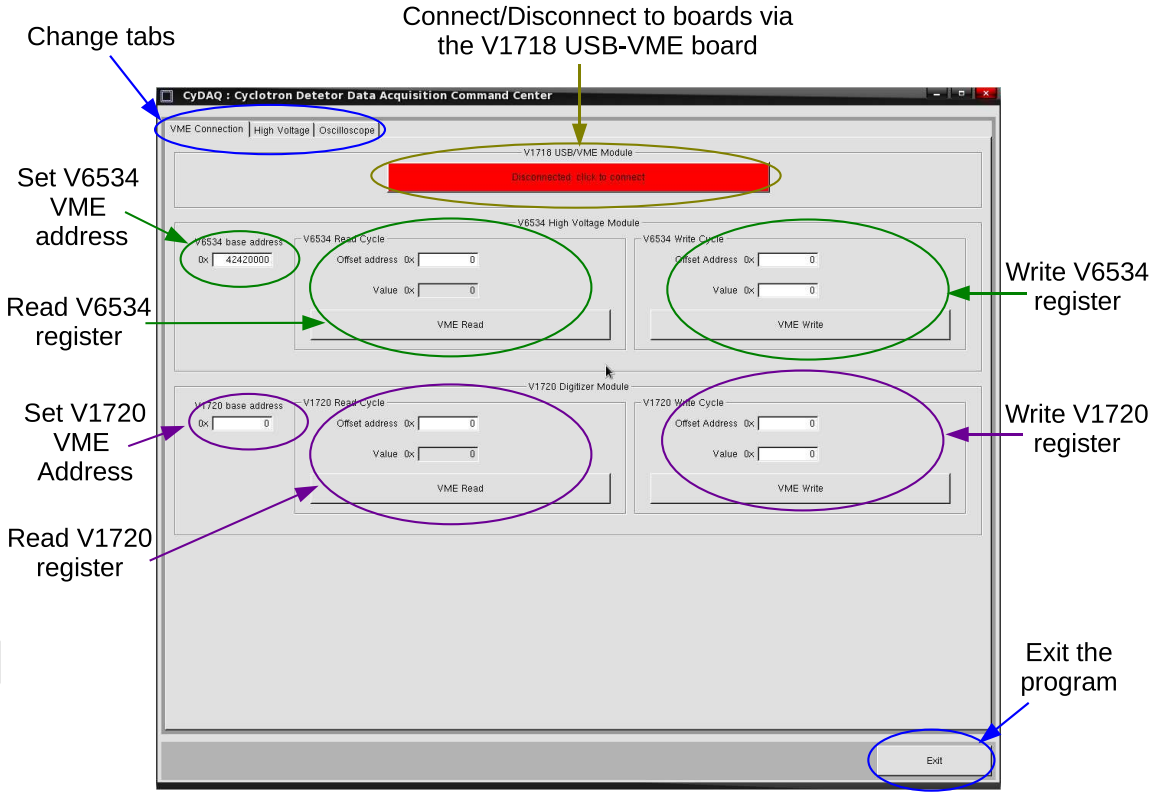
- creates the three objects described above in the active instance of ROOT 's memory. The three class objects, rather than being located on the harddisk are located on the ROOT file (recall that it acts like a unix directory).

- assigns values of the `CyDAQRootMeasParams` object and the `TObjString` by obtaining the values from the appropriate ADAQAcquisition widgets

When the user has clicked the "Data stored while checked" check box having opened a ROOT file, digitized waveforms are actively written to the ROOT `TTree` object. The user can verify this by continuously inspecting the size of the ROOT file, which will grow in size as waveforms are written to the ROOT `TTree` object contained on the ROOT file.

Finally, when the user is done acquiring data and clicks the "Closed ROOT file" button, the three objects are finalized on the ROOT file ("written to the ROOT file" in the ROOT parlance) and a number ROOT protocol operations are performed to finalize the format of the ROOT file before it is closed.

Figure 3.1: The VMEConnection frame as it appears in ADAQAcquisition with annotations describing the general purpose of the containe widgets.



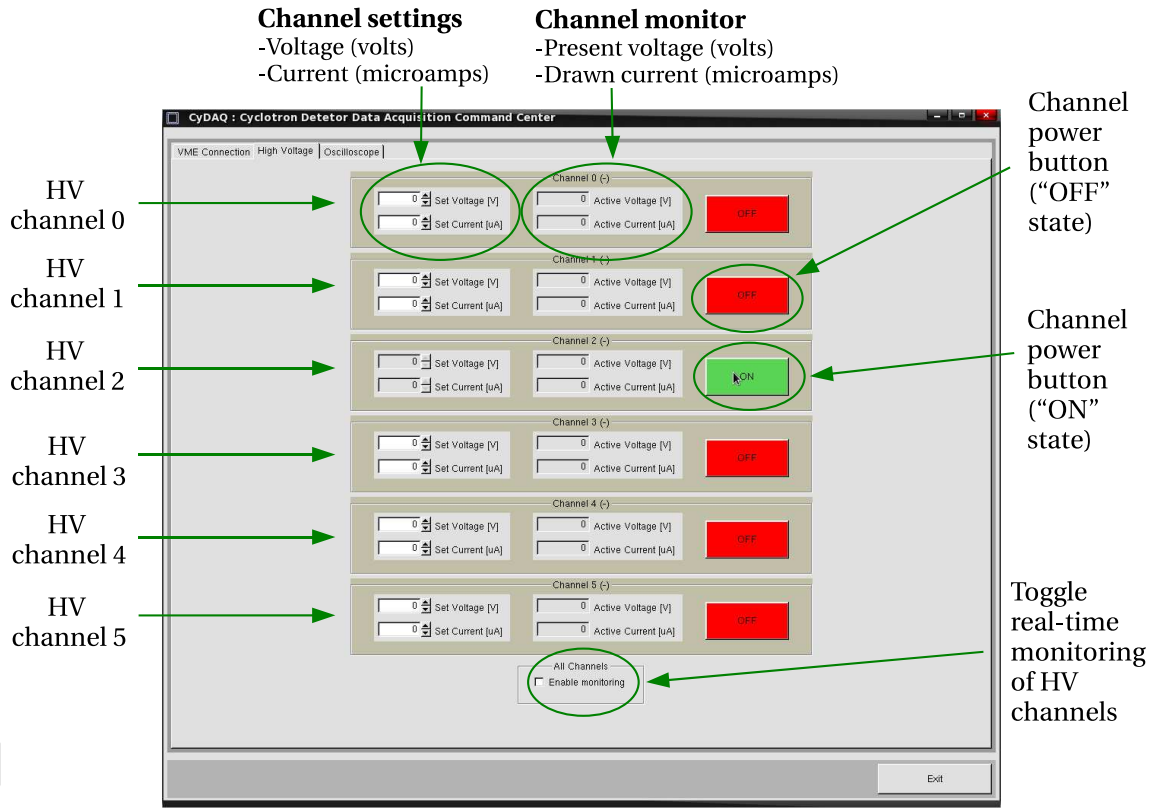


Figure 3.2: The High Voltage frame as it appears in ADAQAcquisition with annotations describing the layout and purpose of the encapsulated widgets.

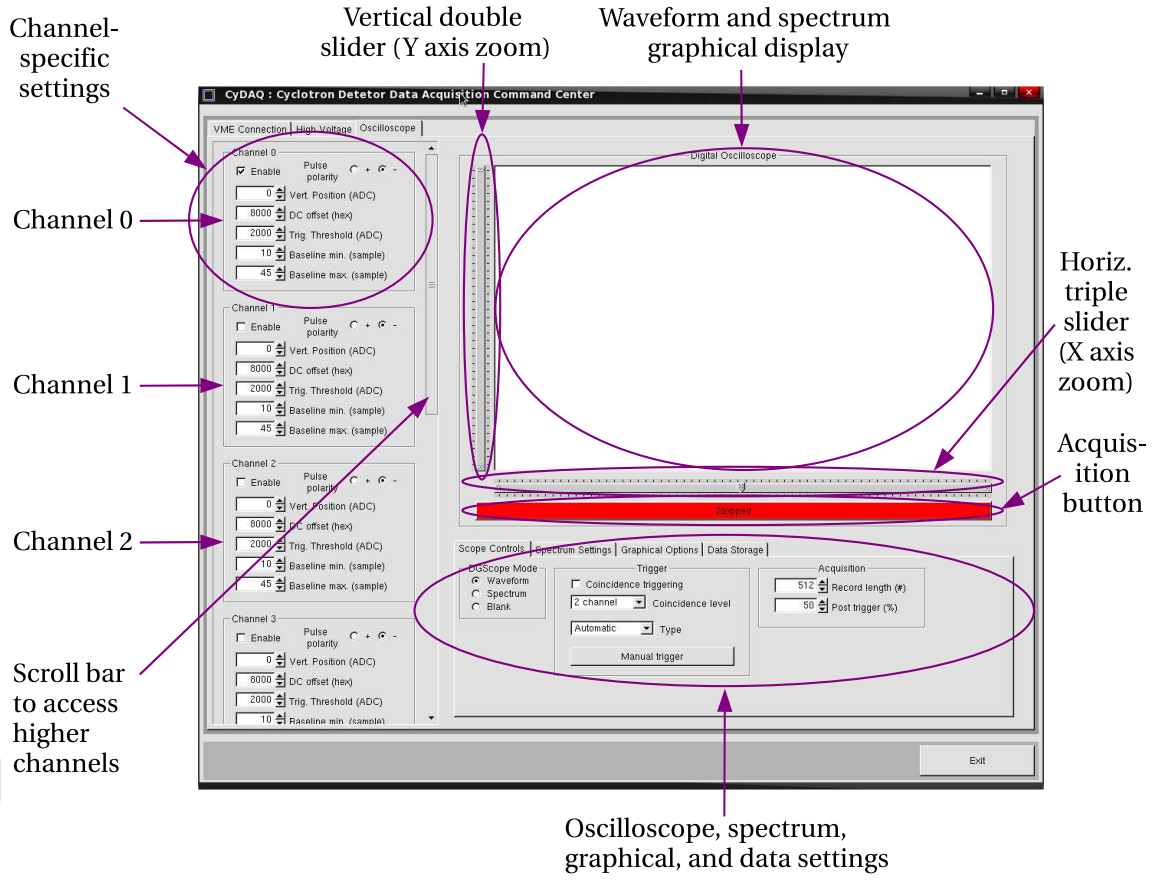


Figure 3.3: The Oscilloscope frame as it appears in ADAQAcquisition with annotations describing the general purpose of the contained widgets.

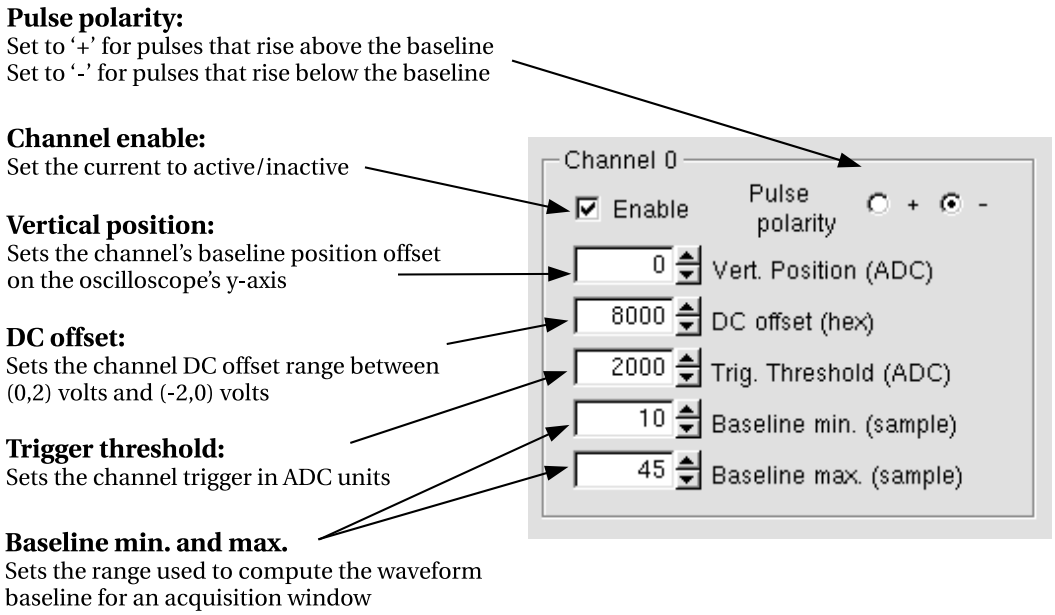


Figure 3.4: An annotated close-up view of one of the eight channel setting subframes contained in the Oscilloscope frame. In this example, each visible widget controls the channel-specific settings for the V1720 digitizer channel 0. A scroll bar is provided on the Oscilloscope frame to enable the user full access to all eight of the channel-specific setting subframes without hampering visibility and usability.

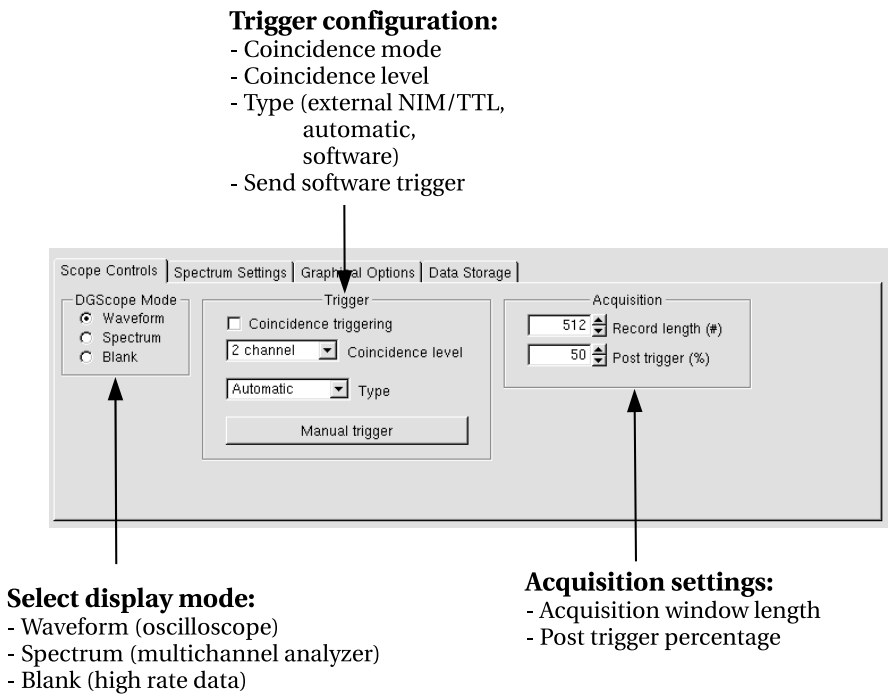


Figure 3.5: An annotated close-up view of the "Scope Controls" subframe contained in the Oscilloscope frame. This subframe contains general settings that affect the global behavior of DGScope, such as what data to display, how to trigger the DGScope, and what the acquisition settings should be.

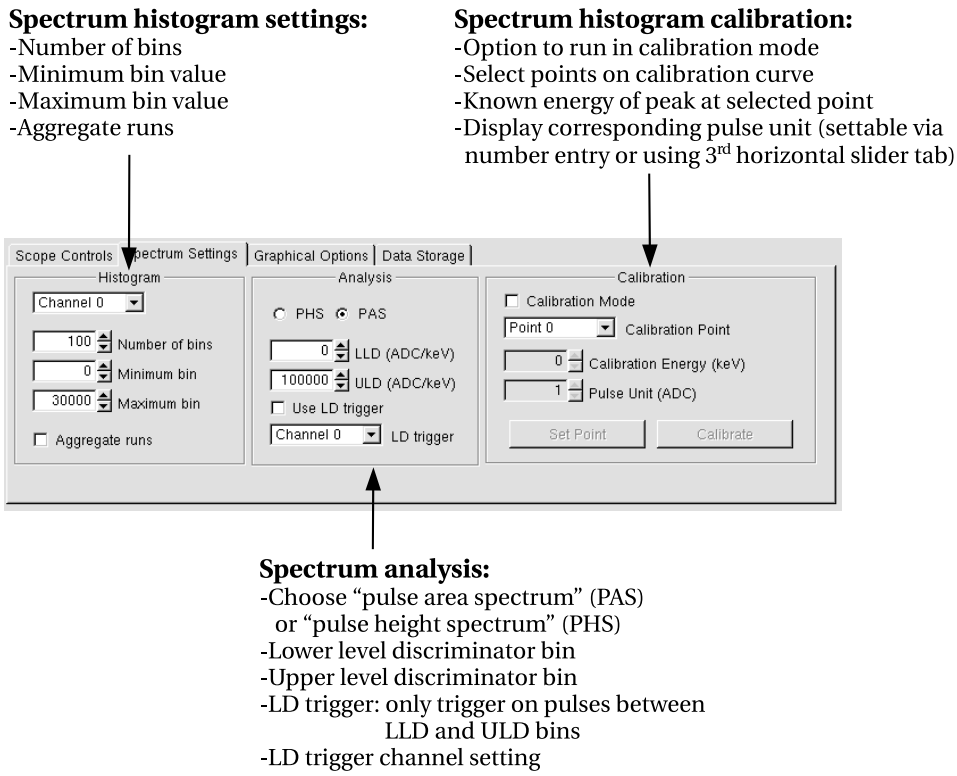


Figure 3.6: An annotated close-up view of the “Spectrum Settings” subframe contained in the Oscilloscope frame. This subframe contains settings for how the acquired waveforms are analyzed and histogrammed to produce a pulse spectrum, as well as calibration settings for the spectrum.

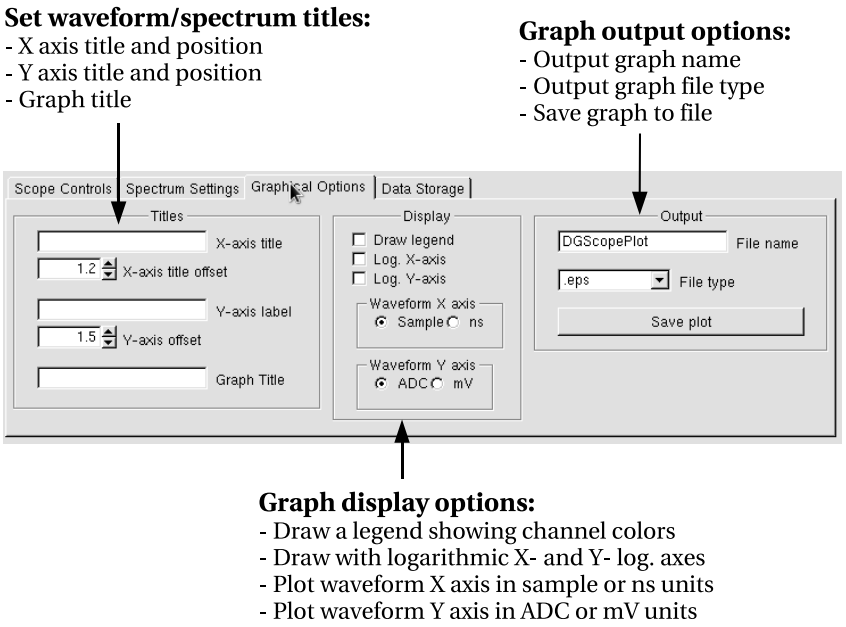
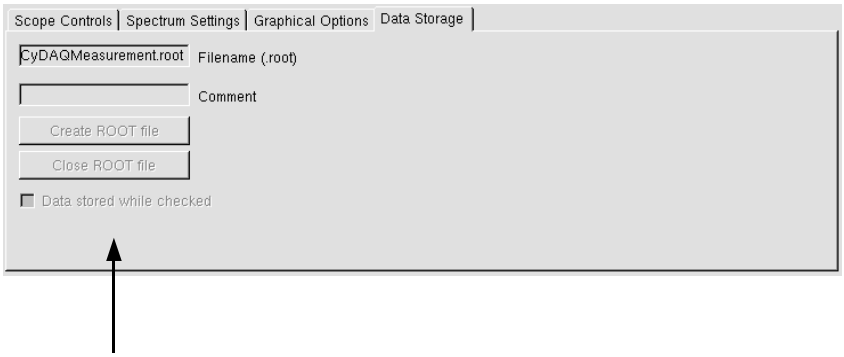


Figure 3.7: An annotated close-up view of the "Graphical Options" subframe contained in the Oscilloscope frame. This subframe contains settings for graphical control of the displayed data in DGScope, as well as options for save images of the displayed data to the hard drive.



ROOT data file options
- ROOT file name
- Comment to be added to ROOT file describing measurement
- Option to create ROOT file (valid when acquiring data)
- Option to close ROOT file (valid when ROOT file open)
- Option to store acquired data in ROOT file (valid when ROOT file open)

Figure 3.8: An annotated close-up view of the "Data Storage" subframe contained in the Oscilloscope frame. This subframe contains settings for persistent storage of the acquired waveform data in ROOT files.

Chapter 4

Tutorial 1: Creating a calibrated energy spectrum

This chapter provides a detailed, step-by-step tutorial that covers using ADAQAcquisition to create a calibrated energy spectrum and output the resulting spectrum to an encapsulated post script (EPS) file. Specifically, the user will learn how to establish a VME connection to the V6534 and V1720 boards, power a standard sodium-iodide detector with the V6534 board, and acquire digital detector waveforms to create an energy spectrum.

4.1 Assumptions

For this walkthrough, a number of *a priori* assumptions have been made:

- The CAEN V1718 driver has been installed on system running `CyDAQRootGUI`.
- The user's environment is configured as described in Section 2.4.
- The CAEN V1718 board occupies slot 1 of the VME8100 crate; the CAEN V6534 and V1720 VME boards have their VME address set to 0x42420000 and 0x00420000, respectively, using the physical potentiometers on each board.
- The system running `CyDAQRootGUI` is connected to the V1718 board via the USB-2.0 cable.
- The NaI-PMT detector requires 1750 volts of *negative bias* bias high voltage and less than 800 microamps of current.
- The NaI detector outputs negative pulses (i.e. pulses that "rise" below the baseline in the range -1 to +1 volts. Typical pulses are on the order of a few hundred millivolts to a thousand millivolts.
- The user has a ^{137}Cs ($E_\gamma = 661.7$ keV) and ^{60}Co ($E_\gamma = 1170$ and 1330 keV) radioactive source available.

4.2 Hardware setup

This section describes the ordered steps for setting up the ADAQAcquisition hardware in advance of running `CyDAQRootGUI`.

1. Power on the VME8100 crate.
2. Connect the V6534 high voltage channel 0 (SHV connector required) to the high voltage input on the base of the NaI detector.
3. Connect the V1720 digitizer channel 0 (MCX jack required) to the signal output on the base of the NaI detector.

4.3 Configure ADAQAcquisition for acquisition

This section describes starting the `CyDAQRootGUI` software, establishing a VME connection to the CAEN V6534 and V1720 boards, and powering the NaI detector using the V6534 high voltage board.

1. Open a terminal and launch the ADAQAcquisition graphical interface by typing "CyDAQRootGUI" at the command line. You should see the graphical interface open to the "VME Connection" frame described in Section **??**.
2. Ensure that the V6534 and V1720 VME base addresses are correctly set (0x4242000 for the V6534 high voltage board and 0x00420000 in this example).
3. Establish a VME connection to the V6534 and V1720 boards by clicking the large, red connect button at the top of VME connection frame. If the connection is successfully established, the button will turn green and the button's text will indicate a successful connection.
4. Use the tabs at the top-left of the GUI and navigate to the "High Voltage" frame. Using the "Set" widgets for V6534 channel 0, assign a value of "1750" (volts) to the "Set Voltage" number entry and a value 'f "800" (microamps) to the "Set Current" number entry. Click the "Enable monitoring" check box at the bottom of the screen; all 6 channel's "Active Voltage" and "Active Current" widgets will turn white, showing they are enabled. Each widget's value should be 0 (or very close to it) since none of the V6534 high voltage channels are on.
5. Click the large, red "Power button" at the rightmost side of channel 0's group frame. The button will turn from red to green and the text will change from "OFF" to "ON" to indicate that V6534 channel 0 is now energized. You should now observe the voltage and current on channel 0 slowly ramp up by obvserving the "Active Voltage" and "Active Current" widgets for channel 0. Note that while the active voltage will rise to the set value of 1750 volts, the active current will rise to whatever current is drawn by the NaI detector's PMT and base electronics.

4.4 Acquiring waveforms

This section describes using `CyDAQRootGUI` to optimally acquire digital waveforms from the detector output pulsest At this point in the tutorial, the ADAQAcquisition system hardware should be producing analog output NaI detector pulses which are being fed into the V1720 digitizer channel 0.

1. Use the tabs at the top-left of the GUI and navigate to the "Oscilloscope" frame. In the channel 0 group frame, ensure that digitizer channel 0 is enabled. Use the rest of the channel 0 default settings.

2. Under the "Scope Settings" subframe in the "Trigger" group frame, set the trigger type to "Software".

3. Click the wide, red button under the blank DGScope canvas. The button will turn from red to green and the text will change from "Stopped" to "Acquiring" to indicate that the V1720 board is now running and waiting to acquire data.

4. Under the "Trigger" group frame, click the "Manual Trigger" button to send a software trigger to the V1720 board. This will tell the V1720 board to convert any analog signal on enabled channels (only 0 in this case) to a digital waveform. You should see an oscilloscope-like plot appear on the DGScope canvas. If the data acquisition gods are on your side, you should see a horizontal solid black line and a horizontal dashed black line appear somwhere in the middle of the plot. The solid line is the baseline for channel 0; the dashed line is the trigger threshold for channel 0. Note that the trigger threshold's y-value on the plot corresponds to the trigger threshold number entry setting in the channel 0 group frame.

5. Using the "Trig. Threshold" number entry in the channel 0 group frame, Set the trigger threshold to be 100-300 ADC units below the channel 0 baseline (because the NaI detector is assumed to have negative pulses polarity). Clicking the "Manual Trigger" button will update the position of the channel 0 trigger threshold line on the DGScope canvas.

6. Click the wide, green acquisition button below the DGScope canvas to cease data acquisition. Under the trigger groupframe, change the trigger type to "Automatic" using the selection box. This tells the V1720 board to trigger channel 0 each time it senses a digitized value that exceeds the channel 0 trigger threshold that was just set.

7. Click the wide, red button under the DGScope canvas to start data acquisition. If, for some obscure reason, the data acquisition gods continue to favor your mere mortal self, you should now see frequent waveforms appearing on the DGScope. Note that only waveforms whose height exceeds the trigger threshold appear. You may adjust the trigger threshold in real time to see changes. Note that high acquisition rates cause a lag between the acquired pulse (relatively low CPU) and plotting the waveform (relatively high CPU). A quick "off/on" cycle of acquisition will usually help when wishing to view the effects of live-time trigger changes.

8. Play around with the other settings under the channel 0 group frame. Note that the min/max baseline settings result in a shaded box drawn over the waveform baseline. This region is used to calculate the channel baseline and must not overlap the pulse in the acquisition window!

9. Play around with the vertical double and horizontal triple sliders that are to the left and bottom, respectively, of the DGScope. Using the vertical slider, zoom in on the channel baseline to a scale of a few tens of ADC. Note the inevitable presence of noise! Using the horizontal slider, magnify certain regions of the pulse by changing the fraction of the total acquisition window that is viewd in the DGScope canvas.

10. Stop the acquisition by clicking the wide, green button.

4.5 Creating a calibrated energy spectrum

This section describes creating adjusting the `CyDAQRootGUI` settings to successfully create pulse spectra ("uncalibrated") and energy spectra ("calibrated").

1. Start waveform acquisition again by clicking the wide, red button. Using the widgets in the channel 0 group frame, ensure the trigger threshold is near (but not too close) to the baseline, i.e above all noise by at least 10 or 20 ADC. Ensure that no saturation of waveforms occurs (except for maybe the occasional large cosmic ray). Ensure that the waveform is comfortably within the acquisition window (if not, change the number of samples in the acquisition window using the "Record Length" number entry or adjust the number of samples that occur after the trigger using the "Post trigger" number entry in the acquisition group frame). Ensure that the shaded baseline calculation region does not overlap the pulses

2. Stop the acquisition.

3. Change DGScope to run in "spectrum" mode by clicking the appropriate radio button in the "DGScope Mode" group frame.

4. Click the "Spectrum Settings" subtab to display the options that effects the spectrum histogram. Leave the settings at their default value. In this example we will create a "pulse area spectrum", where the entire detector waveform above the baseline in the triggered acquisition window is integrated and the final value histogrammed.

5. Start the acquisition

6. After sufficient detector waveforms have been acquired, integrated, and histogrammed, the DGScope canvas should begin to display the histogrammed pulse area spectrum. By default, the spectrum histogram painted in the DGScope canvas is updated every 100 events.

7. By default, the spectrum histgram has 100 bins evenly divided between 0 and 30000 pulse units. If the histogram is cut off at higher energy, stop the acquisition, extend the histogram range by increasing the "Maximum bin" value on in the "Spectrum Settings" subframe to the desired level (possibly increasing the number of bins as well), and restart the acquisition. Repeat until the full expected range of the pulse spectrum is within the bin limits of the spectrum histogram.

8. Expose the detector to the ^{137}Cs source, and ensure that the spectrum contains the expectd features for a NaI detector: photoabsorption peak, Compton gap and edge, Compton continuum, gamma backscatter peak, etc.

9. Click the "Calibration Mode" check box on the right side of the "Spectrum Settings" subframe. A vertical dotted red line should appear in the center of the spectrum in the DGScope canvas. Grab the little slider on the horizontal triple slider just below the DGScope canvas and drag it back and forth. Notice how the vertical red line moves with the spectrum and the "Pulse Unit" number entry updates according to the position of the vertical red line in the spectrum.

10. Place the ^{137}Cs source next to the NaI detector until the photopeak corresponding to the 661.7 keV gamma appears in the pulse area spectrum histogram in the DGScope canvas. Drag the little slider to align the vertical red line with the center of the photopeak. Enter "661.7" in the "Calibration Energy" number entry. Click the "Set Point" button when finished.

11. Remove the ^{137} Cs source and place the ^{60}Co source nera the NaI detector; wait for the photopeaks to appear in the pulse area spectrum corresponding to the 1170 and 1330 keV gammas. Align the red vertical line with the 1170 keV photopeak, enter "1170" in the "Calibration Energy" number entry, and click 'the "Set Point" button. Repeat for the 1330 photopeak.

12. remove the ^{60}Co source and click the "Calibrate" button and uncheck the "Calibration Mode" button.

13. At this point, the values of the histogram bins are in units of keV and the histogram bin limits should be reset to reflect the energy of the incident gammas. Stop the acquisition, change the "Maximum bin" number entry to "1600", and restart the acquisition. The histogram will now bin detector waveforms between 0 and 1600 keV. Expose the NaI detector to the ^{137}Cs and ^{60}Co source, verifying that the three gamma photopeaks appear at the extected energy value in the spectrum histogram (661.7, 1170, 1330 keV). Stop the acquisition.

4.6 Creating an EPS file of the energy spectrum

This section describes outputting the contents of the DGScope canvas, which will contained a labelled gamma energy spectrum, to an EPS file.

1. Using the NaI detector and the ^{137}Cs and ^{60}Co sources, set the desired spectrum limits and number of bins, begin acquisition, and create an energy spectrum. Leave the acquisition running.

2. Click the "Graphical Options" tab to move the the DGScope graphical settings frame. Set the X-axis, Y-axis, and graph titles using the appropriate text entries. Adjust the position of the X-axis and Y-axis positions as needed.

3. Set the output image file name using the text entry at the right of the "Graphical Options" subframe. Note that the file suffix (.eps, .ps, etc) is automatically attached to the file name. Use the selection box underneath to ensure that the ".eps" options is selected to ensure the output file is an encapsulated poscript.

4. Click the "Save plot" button, which will create an EPS file containing the current contents of the DGScope canvas. If the file name "EnergySpectrum" was used, a file name

"EnergySpectrum0.eps" will be created in the directory from whence `CyDAQRootGUI` was executed. Use your favorite postscript viewer to inspect the contents of the file.

5. Click the "Save plot" button again, with the same file name and file type. If the file name "EnergySpectrum" was used, a file named "EnergySpectrum1.eps" will be created in the directory from whence `CyDAQRootGUI` was started. The incrementing integer ensures images with the same file name are not overwritten!

4.7 Conclusion

Congratulations! If you have reached this point without throwing errors or putting your first through the computer screen, you have successfully used ADAQAcquisition to acquire detector waveform data, analyze it to create a calibrated energy spectrum, and saved it into a vector-graphics EPS file, laballed axis and all. The interested user is recommended to repeat this tutorial but explore the numerous `CyDAQRootGUI` settings that were not used in this tutorial.

Chapter 5

Tutorial 2: Saving digitized waveforms to a ROOT file

This chapter provides a detailed, step-by-step tutorial that covers using ADAQAcquisition to store digitized detector waveforms into a ROOT file.

The user is expected to have covered Tutorial 1 before beginning this tutorial, as a number of important skills taught there are critical to this tutorial as well.

5.1 Assumptions

The assumptions for this tutorial are identical to those found in Tutorial 1. Please see Section 4.1.

5.2 Hardware setup

The hardware setup for this tutorial is identical to that found in Tutorial 1. Please see Section 4.2

5.3 Configure ADAQAcquisition for acquisition

5.4 Acquiring waveforms

Setting up waveform acquisition is identical to that found in Tutorial 1. Please see Section ??.

5.5 Saving waveforms into a ROOT file

The section describes the steps to create a ROOT file, store digitized waveforms in it, and successfully close the ROOT file. At this point in the tutorial

1. Configure all waveform acquisition settings to their desired value using the channel 0 specific settings found towards the top-right of the Oscilloscope frame. When done adjusting channel 0's settings, stop the acquisition.

2. Click the "Data Storage" subtab toward the bottom right of the Oscilloscope frame. Note that all the widgets on this subframe are disabled.

3. Start the acquisition. Note that "Filename", "Comment", and "Create ROOT File" widgets are enabled, indicating that these widgets may only be used when acquisition is running.

4. Set the ROOT filename. Note that the user is responsible to providing an appropriate suffix, with ".root" being the recommended value. When done, click the "Create ROOT File" button. This creates and initializes the ROOT file on the hard disk, as well as enables the "Close ROOT file" and "Data stored ..." check box, since the ROOT file is now open and can receive data. Note that at this point no waveforms are being written to the ROOT file even though waveforms are being acquired by `|CyDAQRootGUI`.

5. Click the "Data stored ..." check box at the bottom of the subframe. The digitized waveforms on channel 0 that appear on the DGScope canvas are now being written to the ROOT file. The user may continually view the size of the created ROOT file to convince him- or herself that the ROOT file is receiving data and growing in size.

6. Once it is determined that sufficient data has been acquired (by viewing and being satisfied with a pulse spectrum or completing a 5 minute data acquisition session, for example), uncheck the "Data stored ..." check box. Waveforms have ceased to be written to the ROOT file; the ROOT file is still open and must be closed correctly to ensure the file is not corrupted.

7. Click the "Close ROOT File" button. This performs a number of final, required operations on the ROOT file and closes it.

At this point in the tutorial, a complete ROOT file exists on the hard disk containing digitized waveforms and can be processed with offline data acquisition code. The user may now: create a new ROOT file and repeat the above steps; cease acquisition, modify channel settings and repeat the above steps; or move on to a completely different task.

5.6 Offline analysis using the ROOT file

The ADAQAcquisition user is responsible for developing his or her own code that will be used offline to process the data contained in the ROOT file. To guide the user in creating his or her own analysis code, a data analysis "template" has been created that explicitly shows the user how to:

- create a standalone, C++ and ROOT based offline analysis code
- open a ADAQAcquisition ROOT file and extract the V6534 and V1720 channel parameters
- extract the digitized waveforms into C++ integer arrays for processing

The template is configured to open a ROOT file (the path to the file must be specified on the command line), print the V6534 and V1720 channel parameters, and the cycle through all of the V1720 channel 0 waveforms contained in the ROOT file.

The user should copy the template directory to a new directory, and use it as the basis for his or her own analysis code. All of the critical interactions with the ROOT file that must be performed to extract all stored data are explicitly shown and heavily commented, providing the user with a good understanding of the steps as well as a solid foundation for his or her own analysis code. The template code may be found in the `$CYDAQHOME/analysis/analysisTemplate/` directory.

5.7 Conclusion

Congratulations! If you have reached this point without throwing errors or cursing the author with reckless abondon, you have successfully used ADAQAcquisition to acquire detector waveform data and store it in a ROOT file for permament storage and offline data analysis. The interested user is recommended to repeat this tutorial but explore the numerous `CyDAQRootGUI` settings that were not used in this tutorial, especially acquiring data in coincidence and using the level discrimanator settings as the "trigger" for storing waveforms in the ROOT file.